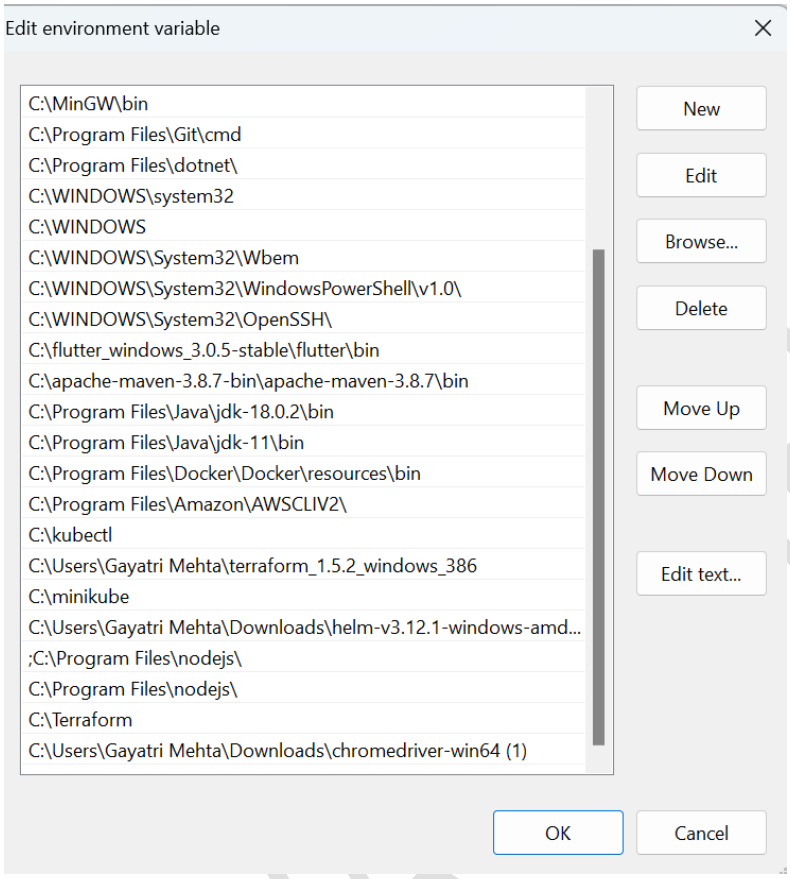
SUBMITTED BY: GAYATRI MEHTA, BATCH 1(NON HONS.)

Lab Exercise 1– Install Terraform on Windows

Step 1: Download Terraform File for Windows.

Step 2: Add Terraform Path to System Environment Variables.



Step 3: Verify Windows Terraform Installation.

```
C:\Users\Gayatri Mehta>terraform -version
Terraform v1.5.2
on windows_386

Your version of Terraform is out of date! The latest version
is 1.7.2. You can update by downloading from https://www.terraform.io/downloads.html
```

SUBMITTED BY: GAYATRI MEHTA, BATCH 1(NON HONS.)

Lab Exercise 2– Terraform AWS Provider and IAM User Setting

Prerequisites:

- Terraform Installed: Make sure you have Terraform installed on your machine. Follow the official installation guide if needed.
- AWS Credentials: Ensure you have AWS credentials (Access Key ID and Secret Access Key) configured. You can set them up using the AWS CLI or by setting environment variables.

Procedure:

Step 1: Create a New Directory

```
PS C:\terraform-scripts> pwd

Path
----
C:\terraform-scripts
```

Step 2: Create Terraform Configuration File (main.tf):

```
main.tf
terraform {
required_providers {
aws = {
source = "hashicorp/aws"
version = "5.31.0"
}
}
}

provider "aws" {
region = "ap-south-1"
access_key = "AKIAVRUVTDBUGK3JSJB4"
secret_key = "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"
}
```

Step 3: Initialize Terraform:

```
PS C:\terraform-scripts> terraform init

Initializing the backend...

Initializing provider plugins...
- Finding hashicorp/aws versions matching "5.31.0"...
- Installing hashicorp/aws v5.31.0...
- Installed hashicorp/aws v5.31.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
PS C:\terraform-scripts>
```

SUBMITTED BY: GAYATRI MEHTA, BATCH 1(NON HONS.)

Lab Exercise 3–Provisioning an EC2 Instance on AWS

Prerequisites:

- Terraform Installed: Make sure you have Terraform installed on your machine. Follow the official installation guide if needed.
- AWS Credentials: Ensure you have AWS credentials (Access Key ID and Secret Access Key) configured. You can set them up using the AWS CLI or by setting environment variables

Procedure:

Step 4: Create Terraform Configuration File for EC2 instance (instance.tf):

```
instance.tf
resource "aws_instance" "My-instance" {
instance_type = "t2.micro"
ami = "ami-03f4878755434977f"
count = 1
tags = {
Name = "UPES-EC2-Instnace"
}
}
```

Step 5: Review Plan:

```
PS C:\terraform-scripts> terraform plan

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the
following symbols:
  + create

Terraform will perform the following actions:

  # aws_instance.My-instance[0] will be created
  + resource "aws_instance" "My-instance" {
      + ami                                  = "ami-03f4878755434977f"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + cpu_core_count                       = (known after apply)
      + cpu_threads_per_core                 = (known after apply)
      + disable_api_stop                     = (known after apply)
      + disable_api_termination              = (known after apply)
      + ebs_optimized                        = (known after apply)
```

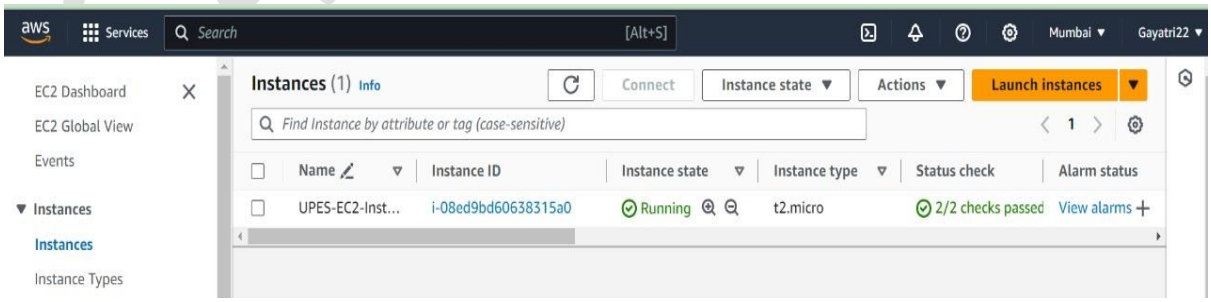
Step 6: Apply Changes:

```
PS C:\terraform-scripts> terraform apply

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the
following symbols:
  + create

Terraform will perform the following actions:

  # aws_instance.My-instance[0] will be created
  + resource "aws_instance" "My-instance" {
      + ami                                  = "ami-03f4878755434977f"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + cpu_core_count                       = (known after apply)
      + cpu_threads_per_core                 = (known after apply)
      + disable_api_stop                     = (known after apply)
      + disable_api_termination              = (known after apply)
      + ebs_optimized                        = (known after apply)
      + get_password_data                    = false
      + host_id                              = (known after apply)
      + host_resource_group_arn              = (known after apply)
      + iam_instance_profile                 = (known after apply)
      + id                                   = (known after apply)
      + instance_initiated_shutdown_behavior = (known after apply)
      + instance_lifecycle                   = (known after apply)
      + instance_state                       = (known after apply)
      + instance_type                        = "t2.micro"
      + ipv6_address_count                   = (known after apply)
      + ipv6_addresses                       = (known after apply)
      + key_name                             = (known after apply)
      + monitoring                           = (known after apply)
      + outpost_arn                          = (known after apply)
```

```
Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_instance.My-instance[0]: Creating...
aws_instance.My-instance[0]: Still creating... [10s elapsed]
aws_instance.My-instance[0]: Still creating... [20s elapsed]
aws_instance.My-instance[0]: Still creating... [30s elapsed]
aws_instance.My-instance[0]: Creation complete after 34s [id=i-08ed9bd60638315a0]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\terraform-scripts>
```

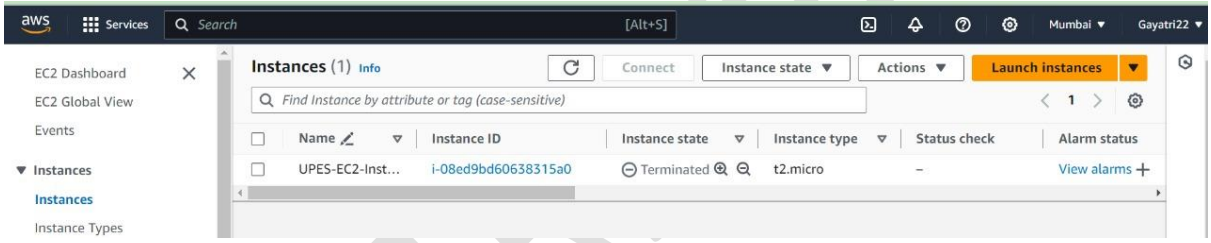
Step 7: Verify Resources:

After the terraform apply command completes, log in to your AWS Management Console and navigate to the EC2 dashboard. Verify that the EC2 instance has been created



Step 8: Cleanup Resources:

```
PS C:\terraform-scripts> terraform destroy
aws_instance.My-instance[0]: Refreshing state... [id=i-08ed9bd60638315a0]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the
following symbols:
  - destroy

Terraform will perform the following actions:

  # aws_instance.My-instance[0] will be destroyed
  - resource "aws_instance" "My-instance" {
      - ami                                  = "ami-03f4878755434977f" -> null
      - arn                                  = "arn:aws:ec2:ap-south-1:381492074600:instance/i-08ed9bd60638315a0" -> null
      - associate_public_ip_address          = true -> null
      - availability_zone                    = "ap-south-1a" -> null
      - cpu_core_count                       = 1 -> null
      - cpu_threads_per_core                 = 1 -> null
      - disable_api_stop                     = false -> null
      - disable_api_termination              = false -> null
      - ebs_optimized                        = false -> null
      - get_password_data                    = false -> null
      - hibernation                          = false -> null
      - id                                   = "i-08ed9bd60638315a0" -> null
      - instance_initiated_shutdown_behavior = "stop" -> null
      - instance_state                       = "running" -> null
      - instance_type                        = "t2.micro" -> null
      - ipv6_address_count                   = 0 -> null
      - ipv6_addresses                       = [] -> null
      - monitoring                           = false -> null
      - placement_partition_number           = 0 -> null
      - primary_network_interface_id         = "eni-00bf6bed0d1073db9" -> null
      - private_dns                          = "ip-172-31-32-17.ap-south-1.compute.internal" -> null
```



GAYATRI

SUBMITTED BY: GAYATRI MEHTA, BATCH 1(NON HONS.)

Lab Exercise 4– Terraform Variables

Objective: Learn how to define and use variables in Terraform configuration.

Prerequisites:

• Install Terraform on your machine.

Procedure:

1. Create a Terraform Directory:
2. Create a Terraform Configuration File.

main.tf

```
provider "aws" {
 region = var.region
 access_key = "AKIAVRUVTDBUGK3JSJB4"
 secret_key = "7cCnN0wzsabhZZI1y6inat9s4bjmsG/EpHopyrlw"
}

resource "aws_instance" "example" {
 ami = var.ami
 instance_type = var.instance_type
}
```

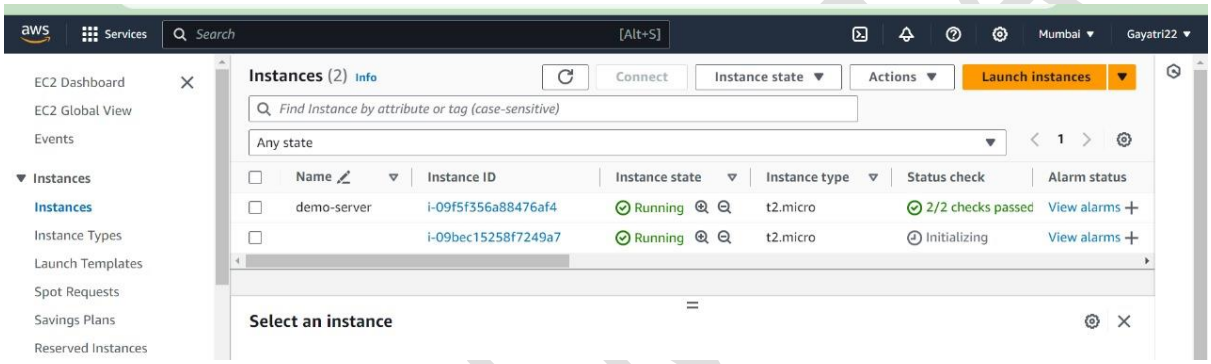
3. Define Variables

variables.tf

```
variable "region" {
 description = "AWS region"
 default = "ap-south-1"
}
variable "ami" {
 description = "AMI ID"
 default = "ami-025680a74fd48deb7"
}
variable "instance_type" {
 description = "EC2 Instance Type"
 default = "t2.micro"
}
```

4. Initialize and Apply

```
PS C:\terraform-scripts> terraform init -upgrade

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Using previously-installed hashicorp/aws v5.34.0

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
PS C:\terraform-scripts> terraform apply

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  + create
```



5. . Clean Up

```
Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\terraform-scripts> terraform destroy
aws_instance.example: Refreshing state... [id=i-09bec15258f7249a7]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  - destroy

Terraform will perform the following actions:

  # aws_instance.example will be destroyed
  - resource "aws_instance" "example" {
```